15.053 Tuesday, April 10

The Network Simplex Method for Solving the Minimum Cost Flow Problem

Quotes of the day

I think that I shall never see
A poem lovely as a tree.

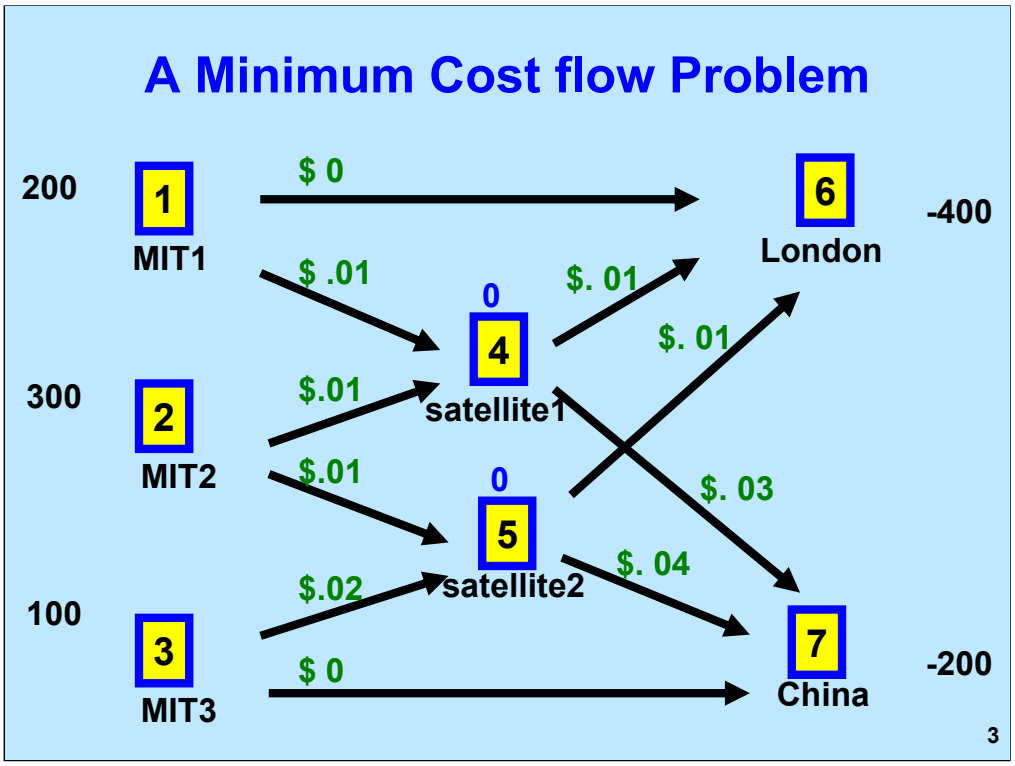
-- Joyce Kilmer

Knowing trees, I understand the meaning of patience.

-- Unknown

Every man has his price.

-- Robert Walpole



Here is the min cost assignment problem from the previous lecture.

The Minimum Cost Flow Problem

- Directed Graph G = (N, A).
  - Node set N, arc set A;
  - Lower bound of 0 on arc (i,j)
  - No upper bounds on arc flows in this lecture
  - Cost c_{ij} on arc (i,j)
  - Supply/demand b_i for node i. (Positive indicates supply)

Minimize the cost of sending flow

s.t. Flow out of i - Flow into i = b_i

$x_{ij} \geq 0$ for all $(i,j) \in A$

4

We will be treating the min cost flow problem in which there are no capacities. The algorithm that we present here readily generalizes to the problem in which there are capacities. But we do not include the generalization here because it adds one extra level of complexity, and it is difficult to follow the first time around.

LP Formulation

Minimize $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij}$

subject to $\sum_{j=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{ki} = b_i, \forall i = 1,\ldots,n$

$x_{ij} \geq 0$

5

Here is an LP formulation of the generic min cost flow problem with no capacities.

The Network Simplex Algorithm

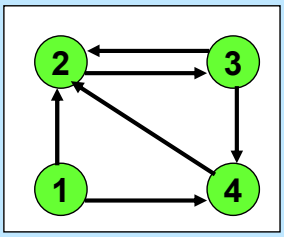
- **We will present the simplex algorithm as it applies to the min cost flow problem**
- **NO TABLEAUS (except next two slides)**
  - **We compute the primal solution directly on the network**
  - **We compute the simplex multipliers directly on the network**
  - **We compute reduced costs directly on the network.**

6

The Network simplex algorithm provides another opportunity to visualize the simplex algorithm. In this case, one can visualize the algorithm in multiple dimensions (that is lots of variables), as opposed to the lectures on geometry where we were restricted to two or three dimensions (that is, two or three variables).

We will not be using tableaus. Nevertheless, we will still compute basic feasible solutions as well as reduced costs. To carry out these computations, we will work directly on the network.

Formulating a min cost flow problem



- x_{ij} = flow in (i, j)
- arc costs c_{ij}
- no arc capacities today
- node supply/demands b_i

Minimize $\sum_{(i,j)\in A} c_{ij} x_{ij}$

x_{12}	x_{14}	x_{23}	x_{32}	x_{34}	x_{42}		RHS
1	1	0	0	0	0	=	b(1)
-1	0	1	-1	0	-1	=	b(2)
0	0	-1	1	1	0	=	b(3)
0	-1	0	0	-1	1	=	b(4)

$x_{ij} \geq 0$

for all arcs $(i, j) \in A$

7

To recall how to compute prices, it helps to write a minimum cost flow problem as a linear program.

Prices and Pricing Out

z	x_{12}	x_{14}	x_{23}	x_{32}	x_{34}	x_{42}		RHS	Prices
-1	c_{12}	c_{14}	c_{23}	c_{32}	c_{34}	c_{42}		0	
1	1	1	0	0	0	0	=	b(1)	y_1
-1	-1	0	1	-1	0	-1	=	b(2)	y_2
0	0	0	-1	1	1	0	=	b(3)	y_3
0	0	-1	0	0	-1	1	=	b(4)	y_4

Reduced Costs of the Arcs

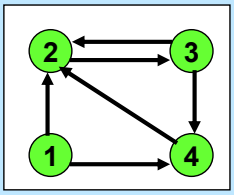
x_{12}	x_{23}	x_{ij}

c_{12}

$- (y_1 \times 1)$

$- (y_2 \times -1)$

$c_{12} - y_1 + y_2$



8

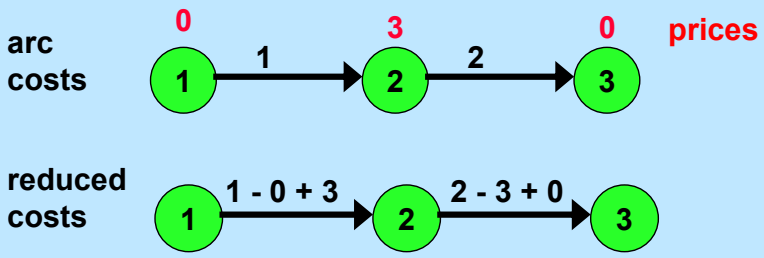
We use the usual rule for pricing out. Note that we have used -1 as the coefficient of z in the z-row, and so the costs are all original costs.

Assuming that the prices are denoted by the vector y, the reduced cost of c_{ij} is $c_{ij} - y_i + y_j$. It's an unusually simple form for a reduced cost.

Pricing Out Again

Let y_i be a "price" associated with node i.

Let $\bar{c}_{ij} = c_{ij} - y_i + y_j$ = *reduced cost* of (i, j) wrt to y.



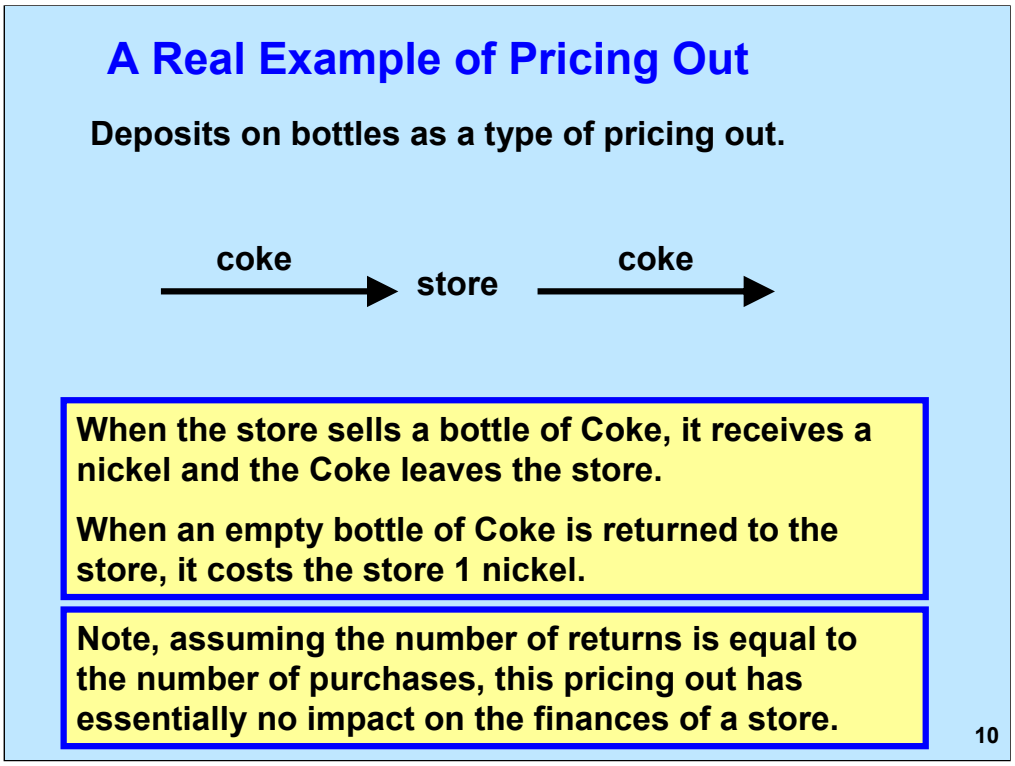
Note: $\bar{c}_{12} = c_{12} + 3$

$\bar{c}_{23} = c_{23} - 3$

Costs are increased coming into a node. They are decreased going out.

9

We illustrate the reduced costs on a network with two arcs, one directed into node 2 and one directed out of node 2. The point of this diagram is that an increase in the price of node 2 leads to an increase of the reduced costs of arcs coming into node 2 and a decrease of the reduced costs of arcs leaving node 2.



The change in reduced costs can be thought of in terms of bottle deposits from a store's perspective. Every bottle leaving the store results in the store receiving a nickel. The cost to the store went down. Every bottle entering the store results in the store having to pay a nickel. The cost to the store went up.

Assume for now that the number of bottles sold is equal to the number of bottles returned. Then the price of a bottle deposit has no net impact on the store's revenue. (Let's also ignore the time value of money and discounting.) If the deposit charge were $.10 per bottle, the store would still have the same amount of money being paid in bottle deposits as it pays in bottle deposits.

Important Fact: optimizing wrt costs c gives the same optimal solutions as optimizing wrt reduced costs $\bar{c}$.

Using reduced costs will change the cost of each solution by a constant.

Recall $\bar{c}_{ij} = c_{ij} - y_i + y_j$

supply
200

Increasing the price of the node from \$0 to \$10 will result in a net decrease in cost of any feasible flow by 200 x \$10. (Why?)

11

As with all linear programs with equality constraints, prices do not affect the optimum solution. A solution is optimum with respect to the original costs if and only if it is optimum with respect to the reduced costs.

This is illustrated by focusing on node 1, which has a supply of 200 units. If we increase the price of node 1 by \$10, then every solution will have a net decrease in cost by \$10 × 200 = \$2,000. Since the cost of each feasible solution is changed by the constant \$2,000, the change in price does not affect which solutions are optimal.

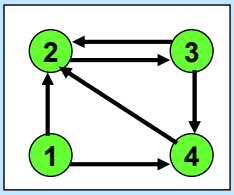
There is always a redundant constraint

z	x_{12}	x_{14}	x_{23}	x_{32}	x_{34}	x_{42}		RHS
-1	c_{12}	c_{14}	c_{23}	c_{32}	c_{34}	c_{42}		0
1	1	1	0	0	0	0	=	b(1)
-1	-1	0	1	-1	0	-1	=	b(2)
0	0	0	-1	1	1	0	=	b(3)
0	0	-1	0	0	-1	1	=	b(4)

Add the last three constraints to the first constraint. What happens to the first constraint?

Can we eliminate constraint 1?

How many basic variables will there be?



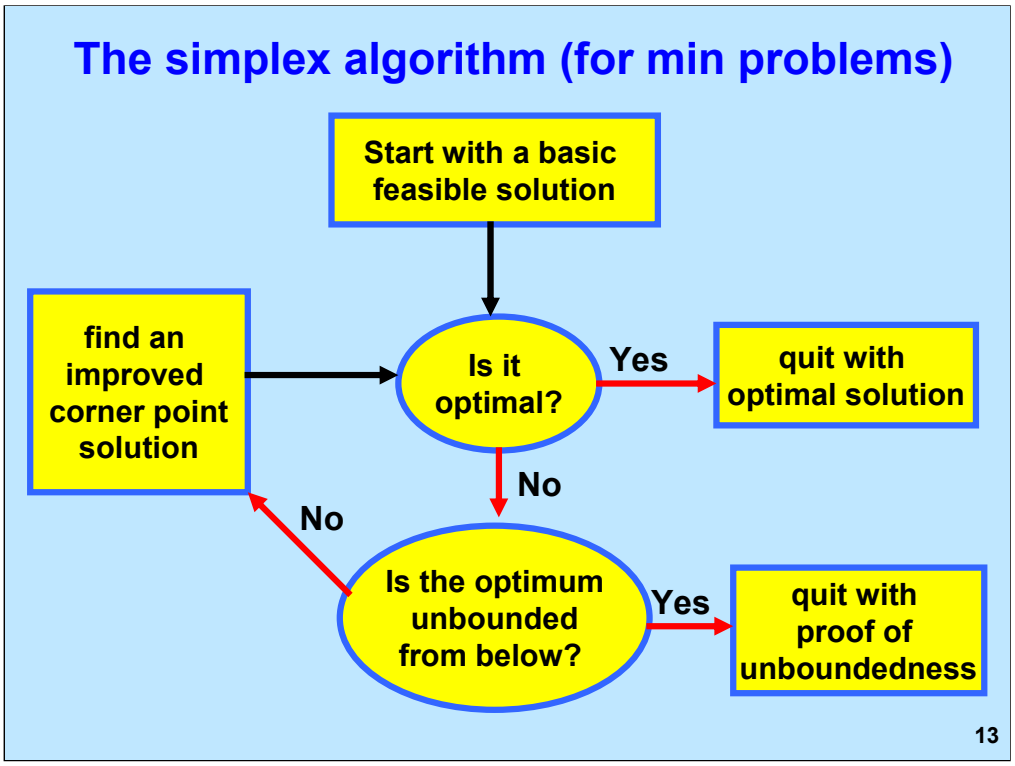
12

In network flow problems, we write out the supply/demand constraint for each node. Each column has exactly one -1 and one +1, with other elements being 0. If we sum the rows, we obtain a value of 0 in each column.

We also assume that b(1) + … b(n) = 0, and so the RHS also sums to 0. This implies that there is a redundant constraint.

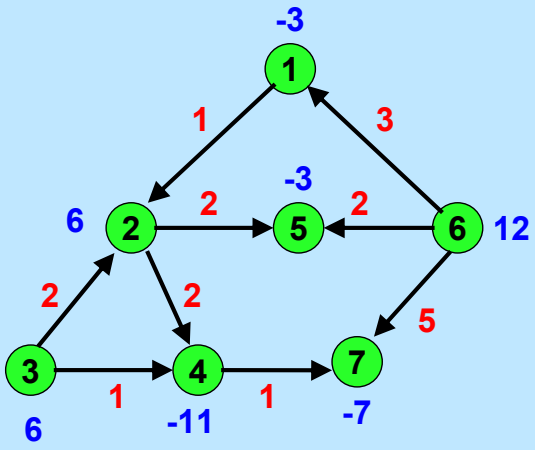
We can eliminate any of the constraints and obtain an equivalent problem. For convenience, we drop constraint 1 corresponding to node 1.

There is usually a price for each constraint. But if we drop constraint 1, there is no need to have a price for this constraint. Equivalently, we can just set y_1 to 0.



We are now ready to explain the network simplex algorithm. We first review a slide used in an earlier lecture on the simplex method. Note that we are minimizing, and so the relevant question concerning unboundedness is whether the optimum objective is unbounded from below.

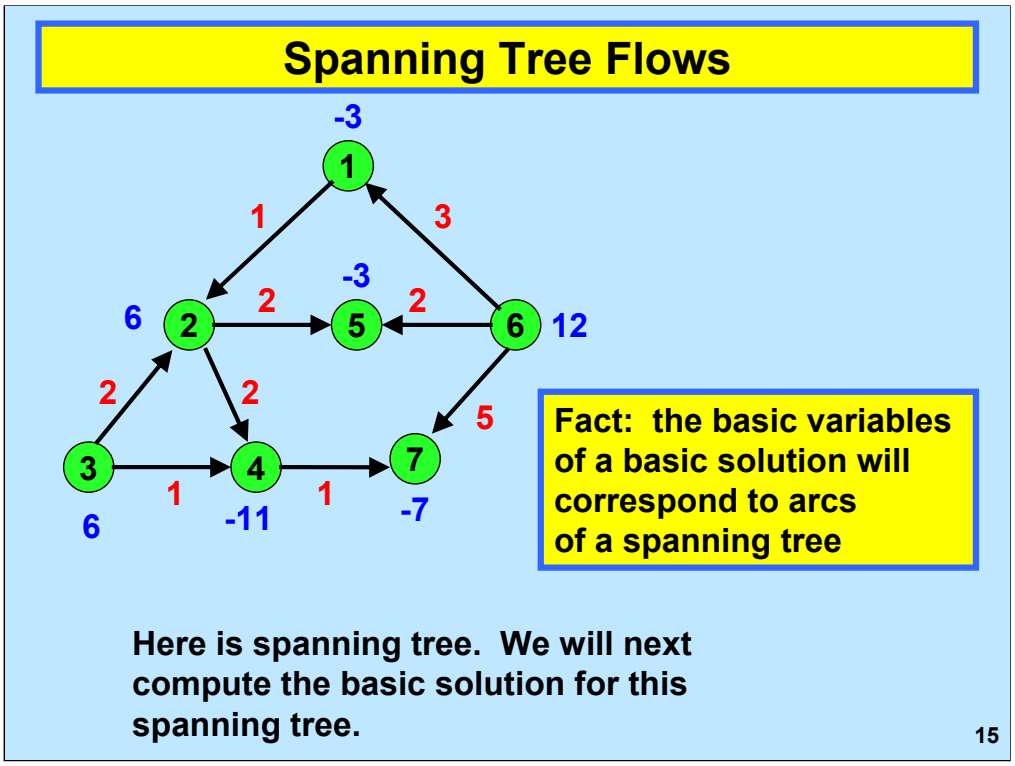
The Network Simplex Algorithm



A minimum cost flow problem.

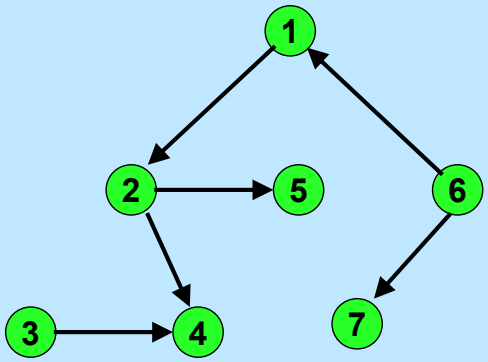
- **The arc numbers are costs.**
- **The node numbers are supplies/demands**

Here is the min cost flow problem that we will solve.



Recall that one constraint can be eliminated, and so a basic solution should have n-1 basic variables. It turns out that there is a 1-1 correspondence between basic solutions and spanning trees. Note that I did not say basic feasible solutions. It is possible that the basic solution associated with a spanning tree will be infeasible.

Spanning Trees and Leaves



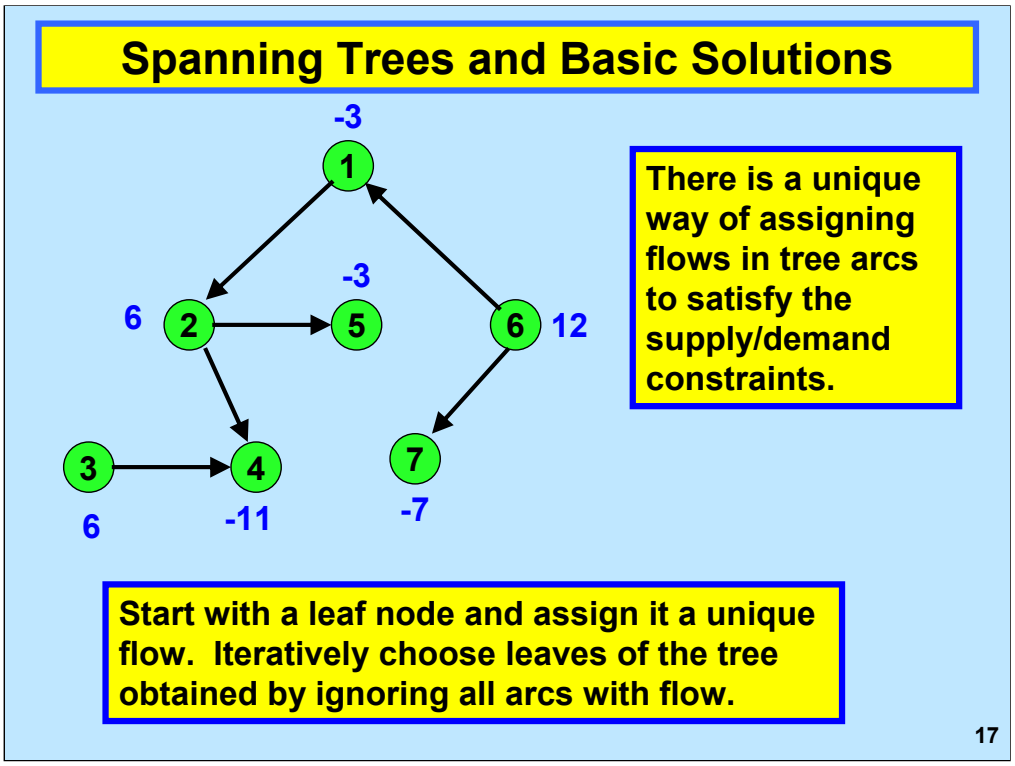
A *leaf* of the tree is a node with one incident arc.

Fact: every spanning tree has at least 1 leaf node. (Otherwise it would have a cycle.)

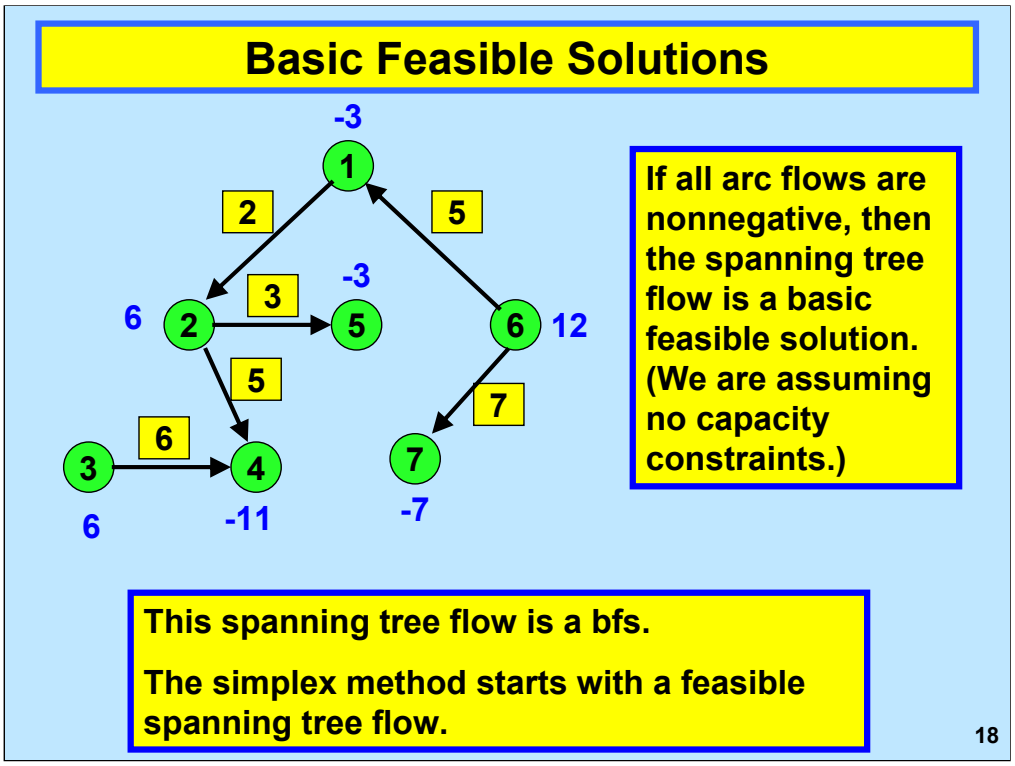
Nodes 3, 5, and 7 are all leaf nodes.

16

We first define leaves, which are needed for the algorithm.

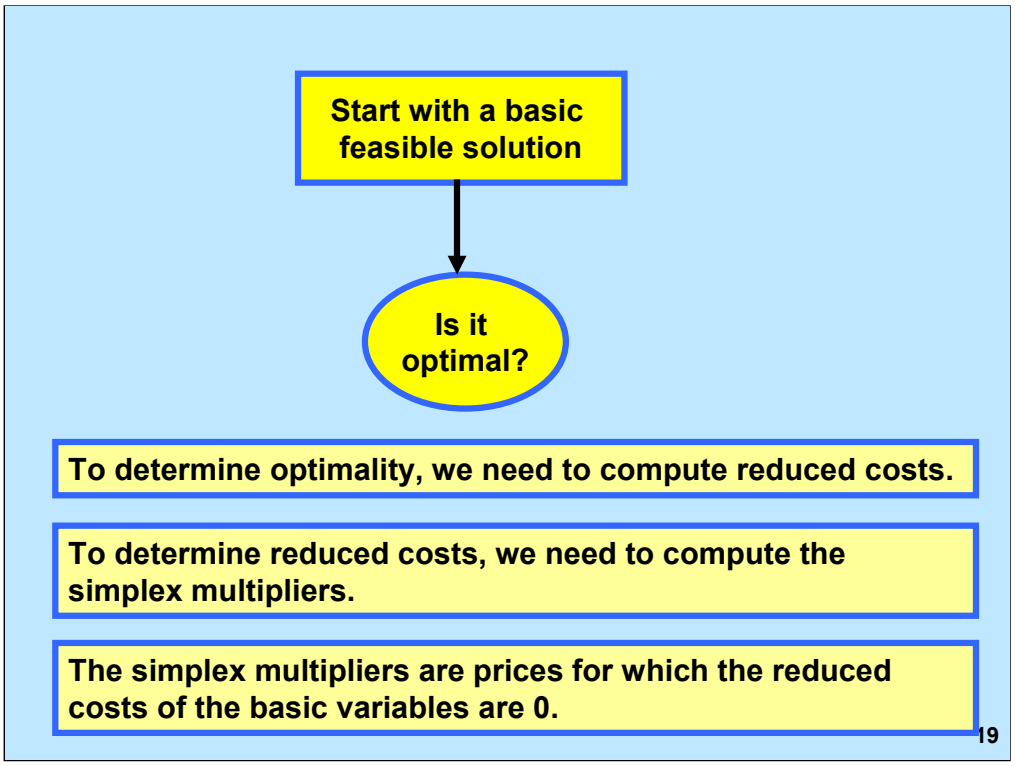


If a node j is a leaf node, then the flow in the arc incident to node j is either b(j) or –b(j) depending on whether the arc is directed from node j or into node u. So, the flow in arc (3, 4) must be 6. If we focus on the green nodes (as per the slide show), then the green nodes always form a spanning tree, and we can always select a leaf of this spanning tree and determine the flow on the arc incident to the leaf.



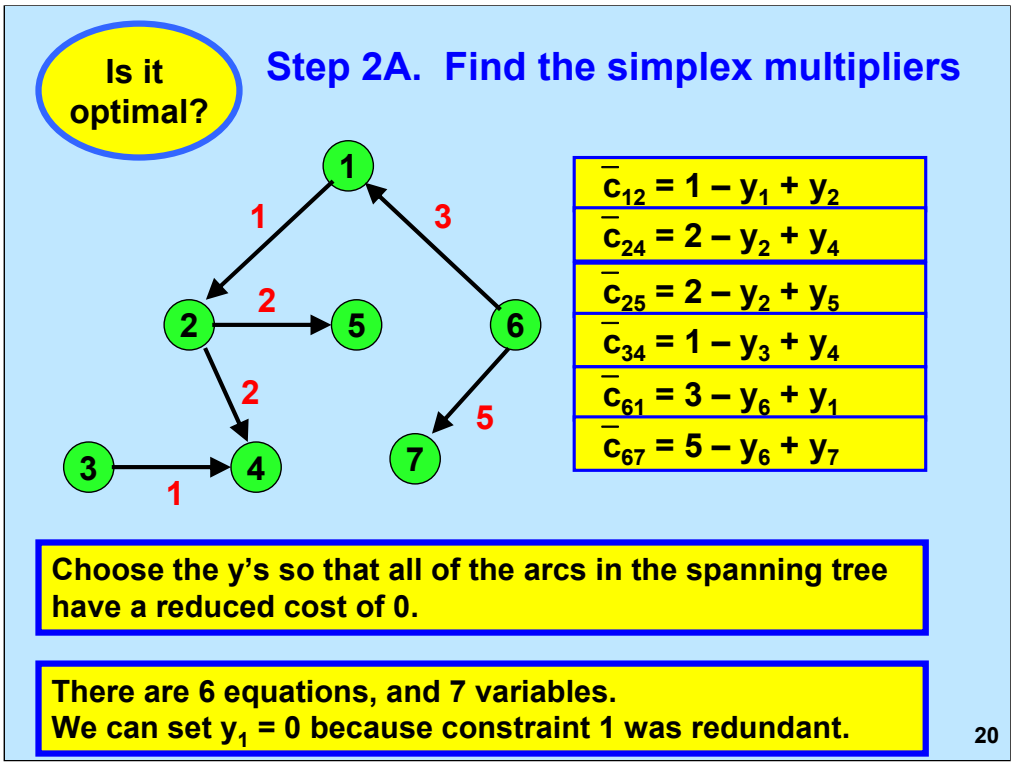
Fortunately, the spanning tree flow was nonnegative, and so it corresponded to a basic feasible solution of the linear program. If one selected an arbitrary spanning tree, one would not expect that the corresponding spanning tree flow would be nonnegative.

As you recall, the simplex method starts with a bfs. So, it is a legitimate question on how to find an initial spanning tree whose flow will be feasible. We will pass on this question for now, just as we passed on it when discussing the simplex algorithm for the first time; however, we will return to the question near the end of the lecture.



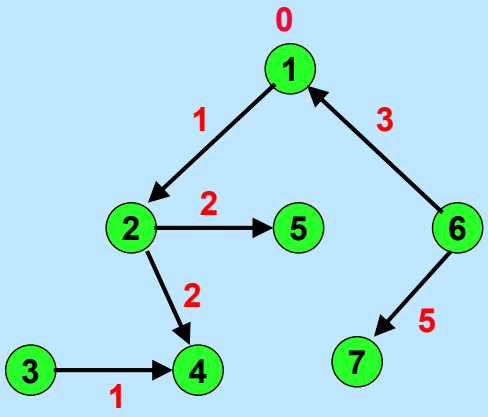
Once one has a feasible spanning tree solution (that is, a bfs), one needs to check whether it is optimal. In the simplex method, this is accomplished by calculating the reduced costs of all of the variables.

We will break up the computation of the reduced costs into two phases. In the first phase, we calculate the simplex multipliers associated with a spanning tree. In the second phase, we use the simplex multipliers to calculate the reduced costs of the nontree arcs (that is, the nonbasic variables).



In general, the simplex multipliers are the unique prices so that the reduced costs of the basic variables are all 0. But for the prices to be unique, we need to deal with the fact that there is a redundant constraint. So, we eliminate constraint 1, and we accordingly set $y_1 = 0$. Once we know that $y_1 = 0$, the other prices are uniquely determined.

Step 2A. Find the simplex multipliers



$0 = 1 - 0 + y_2$

$0 = 2 - y_2 + y_4$

$0 = 2 - y_2 + y_5$

$0 = 1 - y_3 + y_4$

$0 = 3 - y_6 + y_1$

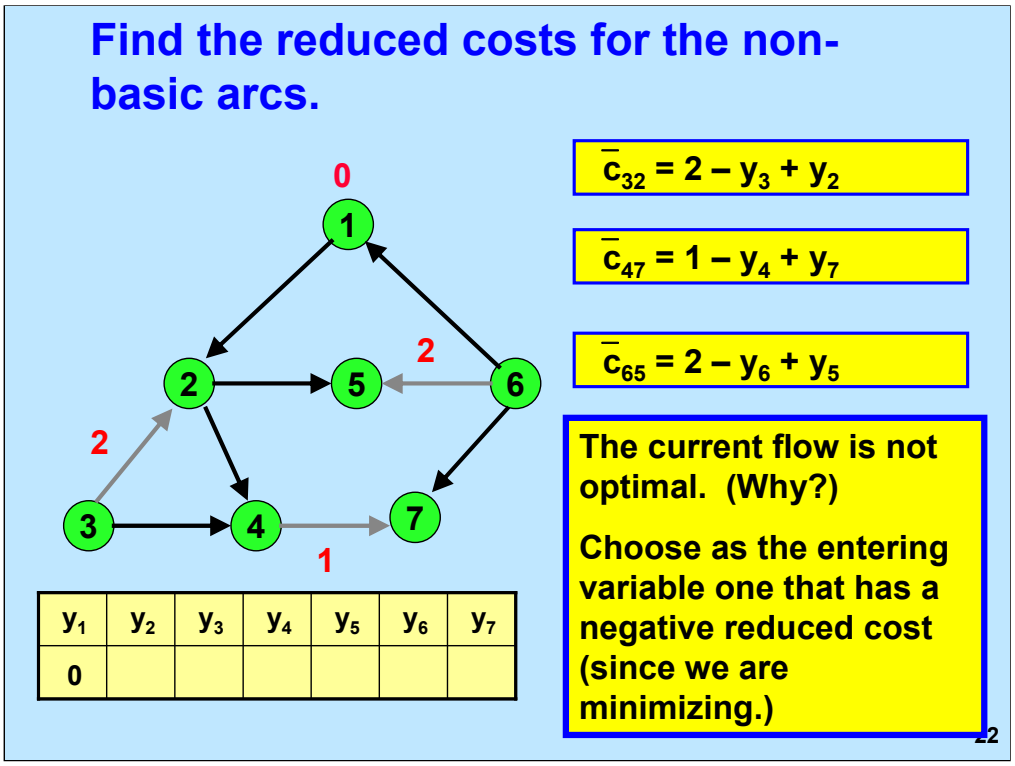
$0 = 5 - y_6 + y_7$

y_1	y_2	y_3	y_4	y_5	y_6	y_7
0						

21

In fact, simplex multipliers are associated with equality constraints of the LP. However, since each equality constraint of the LP is a supply/demand constraint for a node, we associate the simplex multipliers with nodes.

The simplex multiplier for node 1 is 0, by the discussion on the last slide. We now compute the simplex multipliers for all other nodes so that the reduced costs of arcs in the spanning tree are all 0.



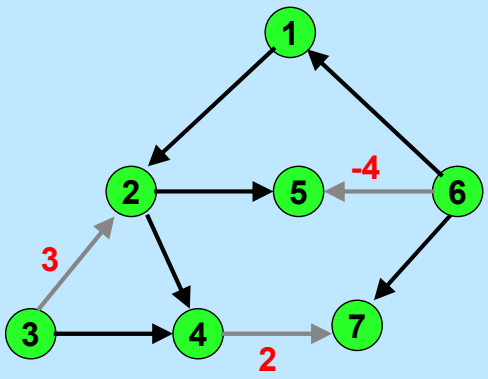
Once we know the simplex multipliers for each node, we can compute the reduced costs of the three nontree arcs. Note that the reduced cost of (6, 5) is negative. This means that if we try to increase the flow on arc (6, 5) we can adjust the flows on other tree arcs and strictly improve the objective function, assuming non-degeneracy.

Accordingly, arc (6, 5) will enter the basis at the next iteration; that is, it will become an arc of the basic spanning tree.

Time for a mental break

Tales of Induhviduals from Scott Adams.

Choose the entering arc



Entering arc for min cost flows: an arc with negative reduced cost.

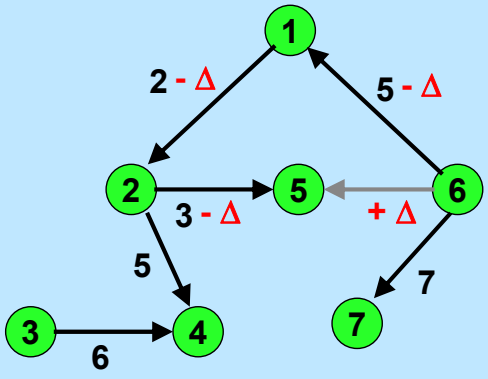
Increase the flow in arc (6,5) and adjust the flows in other arcs to satisfy supply/demand constraints.

Flow will adjust on the arcs of the cycle created by adding the nontree arc to the tree.

24

The following slides carry out the steps of the algorithm, and are self contained.

Send flow around the cycle

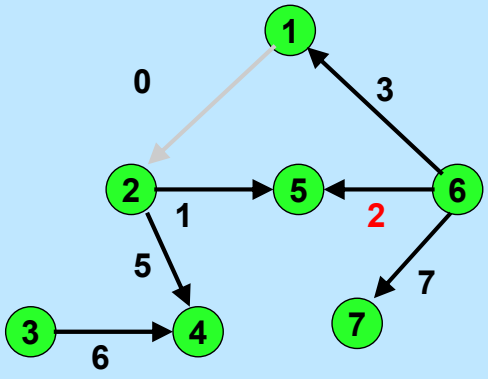


To adjust flows in the basic arcs, modify them along the cycle created by adding (6, 5)

Add Δ to the forward arcs of the basic cycle and subtract Δ from the backward arcs.

Send as much flow as possible until one of the basic arcs has a flow of 0. How large can Δ be?

Determine the exiting arc and the new spanning tree flow.



$\Delta = 2$. Determine the new flows.

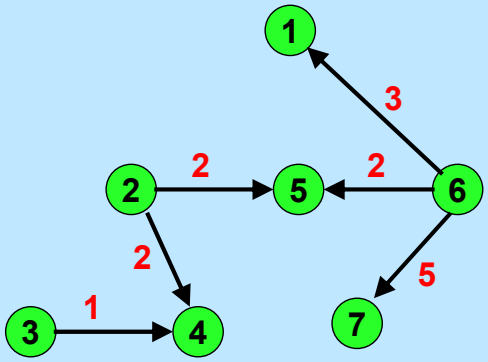
Arc (6, 5) enters the basis. Arc (1, 2) leaves.

At this point, we have a new spanning tree solution. We then compute the simplex multipliers and iterate.

Review of the simplex algorithm

- **Step 1. Start with a feasible spanning tree flow.**
- **Step 2. Compute the simplex multipliers and the reduced cost. If all reduced costs are non-negative, then the flow is optimal. Otherwise, go to step 3.**
- **Step 3. Choose an arc (i, j) with negative reduced cost. Send as much flow as possible around the "basic cycle". Then return to Step 2.**

Choose the simplex multipliers



y_1	y_2	y_3	y_4	y_5	y_6	y_7
0						

$0 = 3 - y_6 + 0$

$0 = 2 - y_6 + y_5$

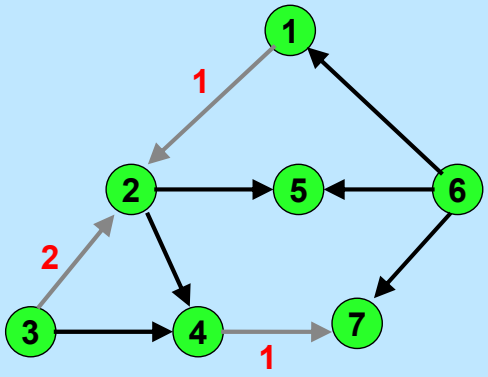
$0 = 2 - y_2 + y_5$

$0 = 2 - y_2 + y_4$

$0 = 1 - y_3 + y_4$

$0 = 5 - y_6 + y_7$

Determine the reduced costs



$\bar{c}_{12} = 1 - y_1 + y_2$

$\bar{c}_{32} = 2 - y_3 + y_2$

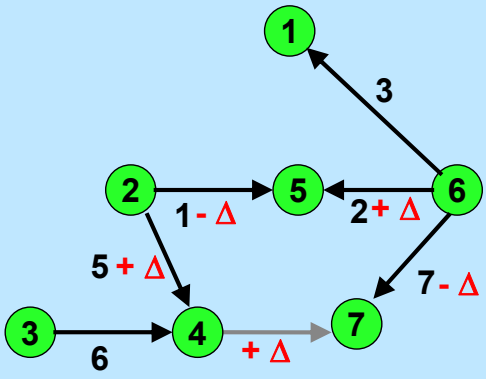
$\bar{c}_{47} = 1 - y_4 + y_7$

y_1	y_2	y_3	y_4	y_5	y_6	y_7
0						

The current flow is not optimal. (Why?)

Choose as the entering variable one that has a negative reduced cost.

Send flow about the basic cycle

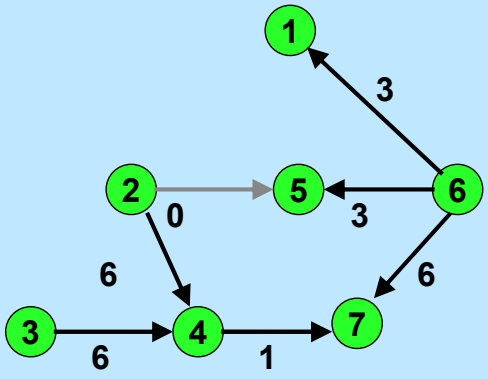


To adjust flows in the basic arcs, modify them along the cycle created by adding (4, 7)

Send as much flow as possible until one of the basic arcs has a flow of 0.

How large can Δ be?

Determine the exiting arc and the new spanning tree flow

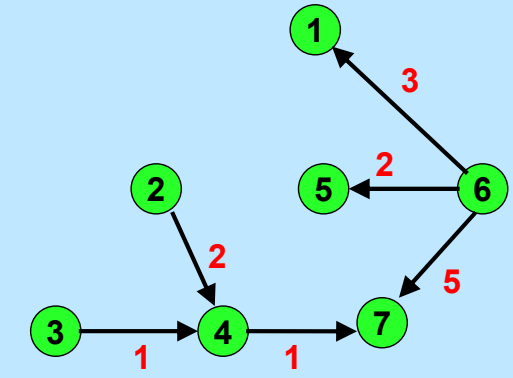


Δ = 1. Determine the new flows.

Arc (4, 7) enters the basis. Arc (2, 5) leaves.

At this point, we have a new spanning tree solution. We then compute the simplex multipliers.

Find the simplex multipliers.



y_1	y_2	y_3	y_4	y_5	y_6	y_7
0						

$0 = 3 - y_6 + 0$

$0 = 2 - y_6 + y_5$

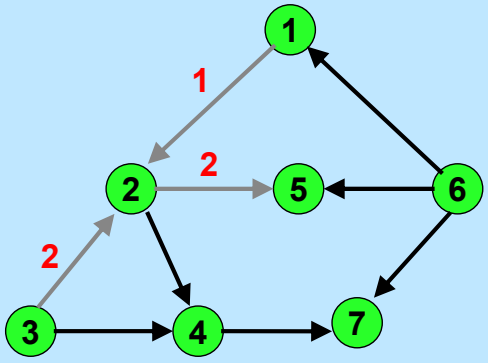
$0 = 5 - y_6 + y_7$

$0 = 1 - y_4 + y_7$

$0 = 2 - y_2 + y_4$

$0 = 1 - y_3 + y_4$

Find the reduced costs



$\bar{c}_{12} = 1 - y_1 + y_2$

$\bar{c}_{25} = 2 - y_2 + y_5$

$\bar{c}_{32} = 2 - y_3 + y_2$

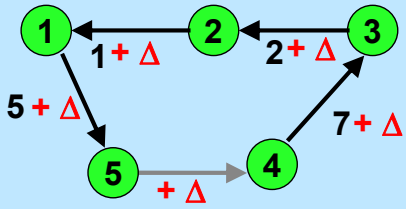
The current flow is optimal.

y_1	y_2	y_3	y_4	y_5	y_6	y_7
0						

33

The algorithm continues until there is an optimum spanning tree flow or a proof of unboundedness. In this case, it ended with an optimum spanning tree flow.

Recognizing when the solution is unbounded from below



If the basic cycle is directed, then the min cost flow is unbounded from below.

34

The proof of unboundedness is when there is a negative cost cycle in which one can send an infinite amount of flow. The solution will never be unbounded from below if each arc has a finite capacity. But we are assuming infinite capacities for this lecture.

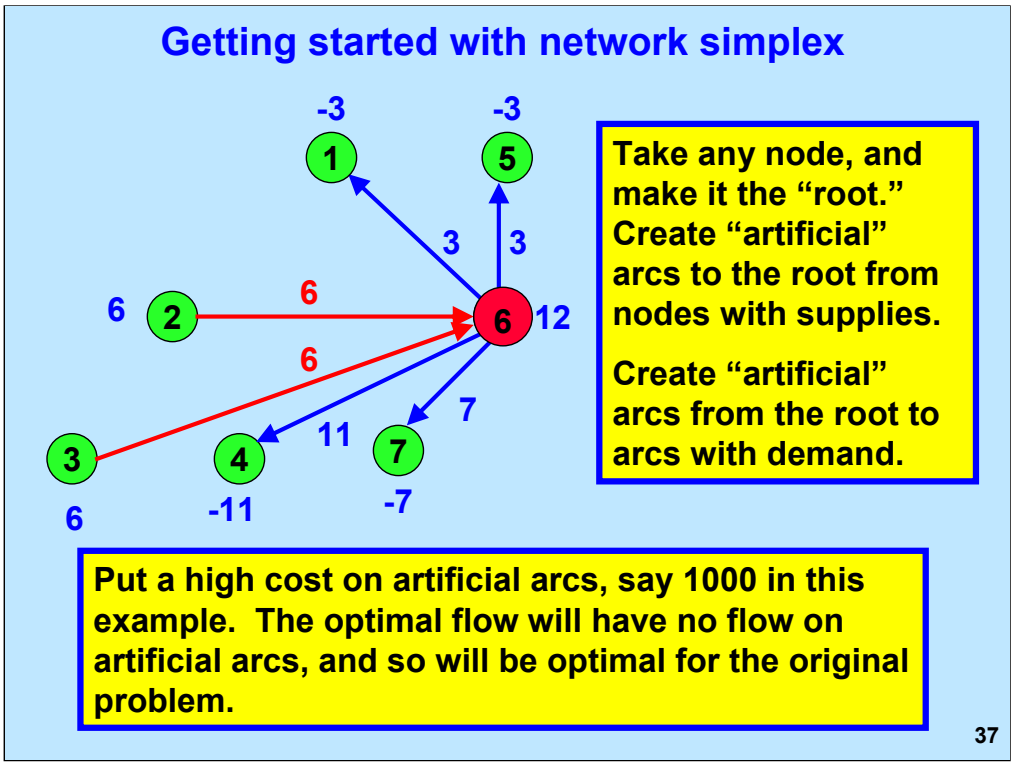
Contrast with the simplex algorithm

- The network simplex algorithm is the simplex algorithm, but without the tableaus
- Bases correspond to spanning trees
- The basic feasible solution is found by sending flow in arcs
- The reduced costs are found by finding the simplex multipliers explicitly.
- The leaving arc is found by sending flow around a cycle.

Of course, this is the simplex algorithm. And so, the issues are how can we interpret the simplex algorithm in this case, and why are there computational differences?

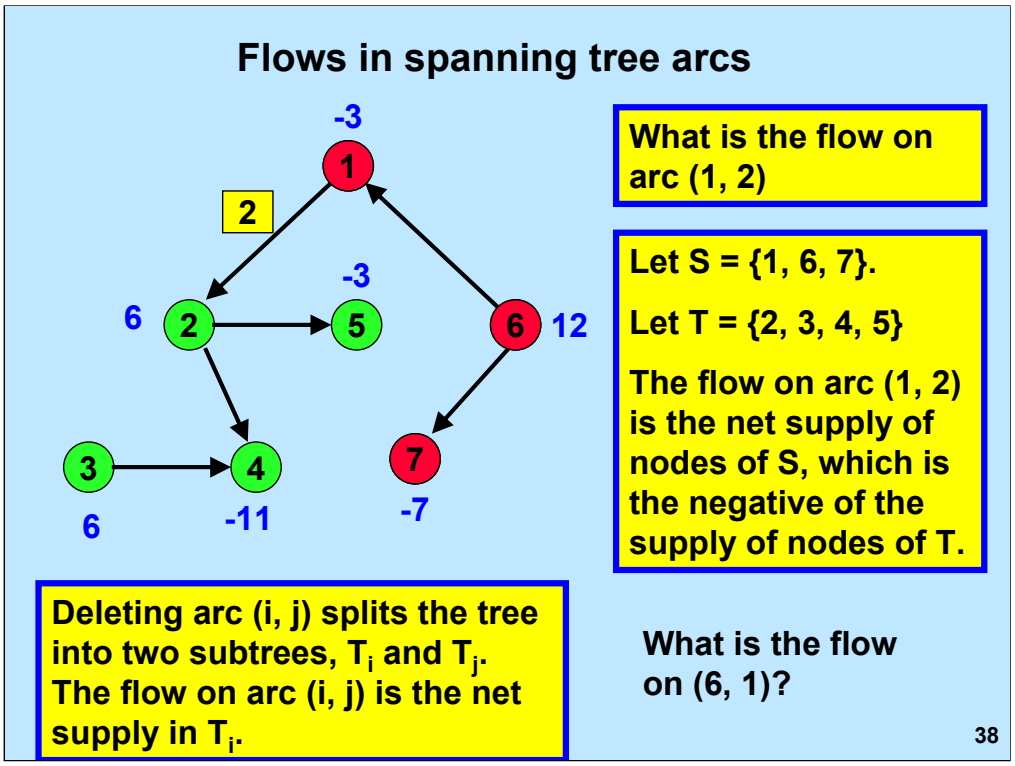
Other properties of network simplex

- How does one find an initial bfs?
- A property of flows in spanning trees
- A property of simplex multipliers
- A property of reduced costs



In the lecture on the simplex algorithm, we described the Phase 1 method in which one tries to obtain a basic feasible solution. If we carried out the Phase 1 method, we would put a cost of 1 on each of the artificial arcs and a cost of 0 on all other arcs.

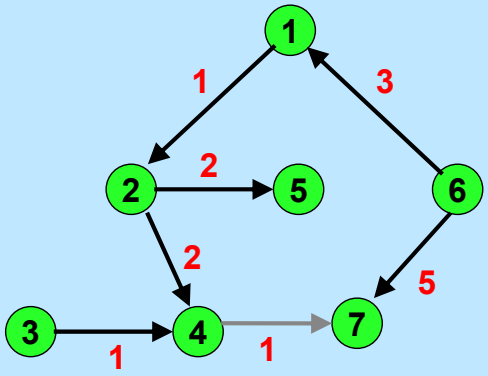
But in this case, we can put a high cost on all artificial arcs in the expectation that none of them would have flow in an optimum solution. In fact, if c* is the largest cost of an arc in the original network, and if one puts a cost of nc* on all of the artificial arcs, one can guarantee that no artificial arc will have positive flow in an optimum solution except in the case that there is no feasible flow.



For a given spanning tree flow, one can compute the flow in each arc of the spanning tree iteratively. However, there is also a more direct method. For example, the flow on arc (1, 2) is the total supply of the red nodes of this diagram. This is because the only way to satisfy the supply constraints at the red nodes is to ship the supply via arc (1, 2). The red nodes are obtained by removing arc (1, 2) from the network and finding all nodes that are connected to node 1.

If one wants to understand an algorithm, it helps to understand it in more than one way. In this case, one can understand the flow in an arc from a global perspective such as above, or by considering the algorithm itself and how flows are allocated.

Costs and Reduced Costs of Cycles



Adding an arc to a spanning tree creates a unique cycle.

cost = 2

The cost of the cycle is the sum of the costs in the forward direction of the cycle minus the sum of the costs in the reverse direction.

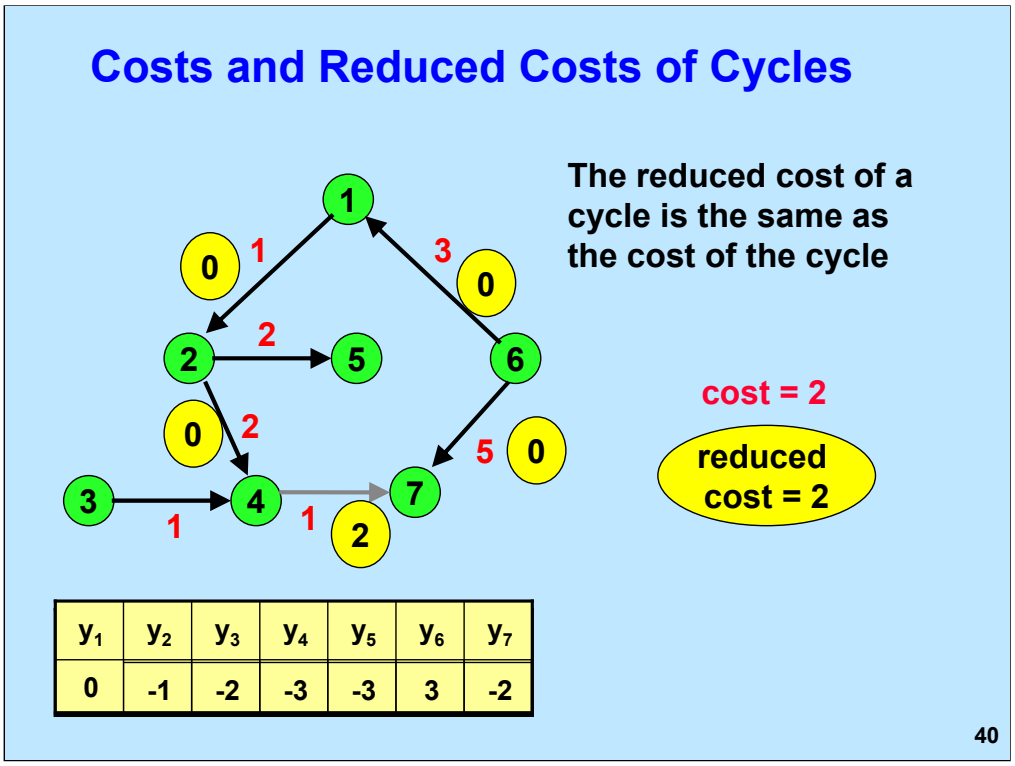
39

It is standard to interpret the cost of a backward arc as the negative of the cost of an arc going in the other direction.

If this point is confusing, think of sending flow around the cycle. To send one unit of flow around the basic cycle, we would need to

1. increase the flow in (4, 7), (6, 7), (1, 2), and (2, 4) by one unit, and
2. decrease the flow in (6, 7) by one unit.

The net impact will be to increase the cost by $7 because of the forward arcs of the cycle, and also to decrease the cost by $5 because we are decreasing the cost in (6, 7) by $5. So, the net increase is $2.



If we choose the prices as the simplex multipliers, then all basic arcs have a cost of 0. Then the reduced cost of the cycle containing any nontree arc (i, j) is also the reduced cost of the cycle.

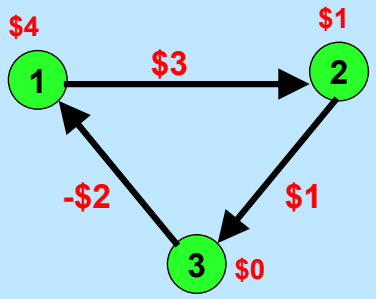
For example, the reduced cost of the cycle 4-7-6-1-2-4 is $2, which is the reduced cost of (4, 7).

We claim here that this is also the cost of the cycle.

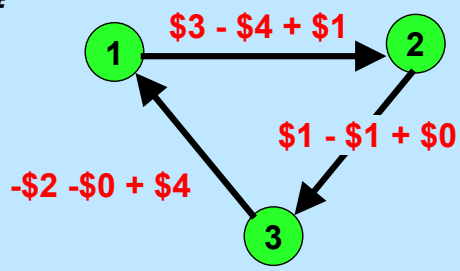
The simplex algorithm works by sending flow around negative cost cycles. Recall that to send a flow around a cycle C is to send increase the flow by Δ in the forward arcs of C and to decrease the flow by Δ in the backward arcs of C.

Reduced Costs of Cycles

- $\bar{c}_{ij} = c_{ij} - y_i + y_j$ for each arc (i,j).



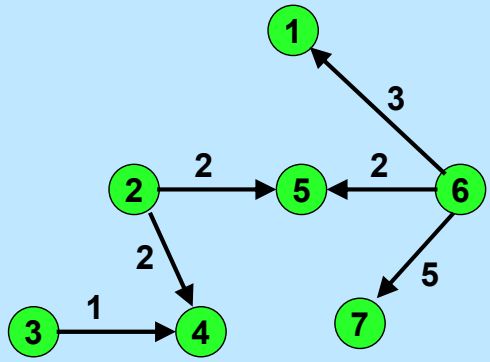
- ***The reduced cost of a directed cycle is equal to the cost of the cycle.***



41

This diagram shows why the cost of a cycle is the same as the reduced cost of a cycle. If we increase the price of node 1 by $k, then the price of the arc entering node 1 increases by $k and the price of the arc leaving node 1 decreases by $k. So changing the price of node 1 (or any other node) does not change the reduced cost of the cycle.

Another method for finding simplex multipliers for spanning tree flows



Assume the multiplier for node 1 is 0.

The multiplier for node j is the length of the path from node j to node 1 in the tree.

e.g., the length of the path from node 4 to node 1 is 5 - 4 = 1.

y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	3	2	1	1	3	-2

42

We compute the simplex multiplier for each node iteratively. But there is also a direct method for computing the simplex multiplier for node j. It is the cost of the unique path in the tree from node j to node 1. (This assumes that the multiplier for node 1 is 0.)

We can illustrate why this works by focusing on node 3. The path from node 3 to node 1 passes through node 4. If we let y_j be the cost of the path from node j to node 1, then we can conclude that $y_3 = y_4 + 1$. This implies that the reduced cost of (3, 4) is 0. Similarly, the reduced cost of every other arc of the tree is 0. But since the simplex multipliers are the unique values that make the reduced costs of tree arcs equal to 0, it follows that vector y is the vector of simplex multipliers.

The significance of the network simplex algorithm

1. It is a lot faster than the usual simplex algorithm. The number of pivots is the same, but each pivot is much faster

2. It gives another view of the simplex algorithm, and its operations.

3. It shows how network algorithms can be much faster.

And now, it's time for …..